CARPARK – A Low-Cost, Implementable Smart Parking Solution for University Campuses

Joshua Benjamin
BNJJOS003@myuct.ac.za

Thabo Kopane
KPNTHA001@myuct.ac.za

KEYWORDS

UCD; IoT; Smart Parking

1. PROJECT DESCRIPTION

The University of Cape Town (UCT) has about 18 000 undergraduate students spread out on five campuses, namely; Hiddingh campus, Middle/law campus (includes main administration office), Lower campus (music and theatre schools and extended administration), health science campus and Upper campus. Upper campus is the main campus, the campus which houses most of the academic departments at UCT.

The UCT website states that parking usage has increased over the last few years because there are more students with cars and students also spend longer times on campus. The number of students needing parking has been increasing rapidly as the website states, but the amount of available parking spots has largely remained the same. The limited parking spots has meant that areas that are inappropriate for parking are continuously being used for parking by students.

The usage of cars for accessing the university campuses will increase for the foreseeable future, because residency closer to campus is becoming more and more expensive. The University has stated that building new bays for parking or creating new parking areas is not a task that is attainable financially at the moment.

This project will attempt to ease the user's experience of parking on campuses, taking into account all of their frustration, in order to develop a system for the users. The idea is that this is a system that is immediately implementable on campus, following the success of the project.

2. PROBLEM STATEMENT

Parking on UCT campuses is a difficult and inefficient process, for both students and academic staff. Low-cost and low-maintenance packaged solutions are not available for our specific campuses. In literature, smart parking solutions are often designed for cities or towns.

Parking on UCT campuses, the upper campus in particular, is an arduous task. Students and academic staff don't know which parking lots have spaces available, so they often spend time trying different bays to try and find parking.

3. AIM

3.1 The Solution

The aim of the project and ultimately the final solution is to create a low-cost, fully implementable smart parking solution for UCT Campuses and any other applications or environments with similar requirements. The aim is to produce a physical module that will display information on campus as well as an accompanying app that will display information to students and staff off-campus at all times.

3.2 What are the possible impacts of our work?

The goal of the system is to make finding parking on campus during peak times easier and more efficient. If the system is well received and utilised, we hope to deploy the app in more parking bays on upper campus and hopefully other campuses at UCT. The project aims to prove that a parking system can be low cost and effective and we hope that other Universities throughout the country that may be struggling with the same problem can adopt the solution for their own usage.

UCD is not a design philosophy that is used often in Software engineering and through this project we aim to highlight that UCD is important during software design.

4. RESEARCH QUESTIONS

4.1 App (Thabo)

Can a mobile application, designed following the UCD methodology, help students and staff use other under-utilised parking bays more?

4.2 Arduino (Joshua)

Is there a low-cost, easy-to-implement and reliable Smart Parking system that can be installed on UCT Campuses to monitor parking lots?

4.3 Both

Can we monitor parking lots in order to divert traffic away from busy parking lots and utilize the less-used parking lots?

5. PRIOR WORK

5.1 Thabo

User centred design as a design philosophy emerged out of the postmodernism [1]. Designers were shifting away from designing

objects or items for form – or that they were shifting towards 'function follows form'[1]. Designing for the users and not inventing users for items is the philosophy at the heart of User/Human Centred Design.

Most of the research articles that involve parking systems don't include the end users during the development phase. Most of the users will interact with the system via the mobile devices, but usually the app isn't a focus on the research, it is only an afterthought [3]. Systems that do focus on the design of the app often neglect potential users of the app during the design phase [4][5].

There are some potentially very useful features that one could include in the app design that could complement and increase the effectiveness of the sensor. One of the features is a crowdsourcing feature [6], where users are able to declare within an app where parking is available. The feature is mentioned quite often in the app-centric research articles such as Grazioli et al [4]'s paper on collaborative mobile application and advanced services for smart parking. It is potentially a useful feature that, if designed properly, could give very effective information to users.

5.2 Joshua

As has been shown through research in the past is that potential parkers spend a significant amount of time searching for parking spots in their pursuit for parking, which often leads to frustration [7]. It has also been noted that the provision of parking information to people enables them to make more informed parking decisions themselves [8], which only adds to their user experience.

Whilst there has been significant work in the area of Smart Parking solutions in the past, a large set of these designs have been for commercial purposes [3] [9], such as can be commonly found in modern shopping centres. In most of these cases, there are mass infrastructural changes required, which is not a viable option in all environments which would want to be monitored.

A various group of sensors has been used in previous projects such as Infrared Sensors [10], Electromagnetic sensors [11] and the more predominantly used, Ultrasonic sensors [11]. Such is the case that a lot of the work has been done with these sensors, but as is often the case, they do not allow for a simple solution, but rather focus on a grand solution or failing that, a proof of concept.

As is evident by the research conducted, there is room for a solution to be developed that abides by the basic requirements of being low-cost and easily implementable, but none have been done that match what is expected of our system. What is also noticeable is the failure of integrating a mobile app with either a pre-existing system or a new system, which we believe to be a possibility and therefore the scope for our project.

6. PROBLEM APPROACH

The project is a software engineering project, it is meant as a proof of concept for the use of smart parking solution to solve the problem of limited parking. We are going to design an Arduino sensor count cars going in and out of a parking bay and we will also design a mobile application that will display aggregated information from the sensor.

6.1 Requirements and Context

For the project to proceed successfully, the following requirements need to be met.

- A survey to understand experiences of parking on campus, and whether there is a need for the parking solution.
- Design a prototype of an Arduino sensor that scans cars going in and out of a parking bay.
- Design a mobile application using User-Centred Design methods, the application should display information from the Arduino sensors on where available parking is.
- Create a server that stores information from the sensor(s), and the application should be able to read from the server.
- Validate the accuracy of the sensor, including, but not limited to the edge cases.
- Validate whether the information in the server matches the sensor information and the information displayed on the mobile device
- Validating the lag of updates from the sensor to the server and to the smartphone.

6.2 Who are the users and clients, and what is their needs?

The project is influenced by the need to solve a local problem on UCT campuses, the potential solution is going to be designed with scalability in mind. Universities that have similar problems and similar parking bays to UCT may be able to use the solution. Because it is primarily a problem-solving project, it is also primarily a software engineering project and thus stakeholders and potential end-users and their needs need to be stated in advance.

The main target users for our software engineering project are students and staff members who drive to and park on campus daily. Parking usage has increased over the years while the amount of available parking bays or spots has not changed. During peak hours, parking utilisation in some of the main parking bays campus, upper campus to be exact, is often high. This would lead students to park their cars at unsuitable areas.

6.3 Methods

As we stated above, we are going to build a sensor for scanning cars, and a mobile device for displaying parking availability. We are going to use a server to integrate the two items.

6.3.1 App

We are following the User/Human Centred Design (UCD) methodology. We will integrate UCD with the Agile Development methodology.

The UCD methodology means that we are including end-users in the design and development phase.

For the development of the application, we are going to have three iterations; a paper prototype, an interactive prototype and the application prototype. The prototype is for both IOS and the Android platforms.

The stages we will follow for development of the app are as follows:

1. Find out who are the potential users of the application and their needs.
2. Investigate the context of use of the application.
3. Gather requirements based off of the usage context.
4. Design the prototype (start with paper, then interactive, then build a first iteration)
5. Evaluate the prototype with potential end users
6. Collate the feedback with the end-users
7. Repeat from step 4 until physical application is designed

Before we conduct the final usability testing of the code, we will collect at least three end-user feedbacks, one for each of the iterations (paper, interactive and first app iteration). The first survey will guide us during requirements gathering, and we can decide on what features to focus on.

The final step after the first app iteration and user testing, to create a minimal viable product with addressed changes from the user feedbacks.

The development and testing of the application will be divided into four different sessions. For each session we do need users for user testing of our prototypes. We want to use a user profiles or personas to decide on users. We plan to recruit users through the Department of Student Affairs, Computer Science mailing lists and posters (with QR sign-ups)

1. The User Survey
    - We want to find out about our users through Online survey
    - Guide with requirements gathering
    - A minimum of 30 users
2. The paper prototype testing
    - Personas and user profiles to narrow users
    - Paper prototype
    - 2 Contextual inquiries
    - One focus group (with 5 users)
    - Recruit potential users with the help
3. The interactive prototype
    - Interactive prototype with mock-up tool.
    - Completing interactive prototype with online survey and 5 real life users
4. The first app Iteration
    - Final iteration
    - Five different users for different contextual inquiries
    - To simulate a task of looking for parking

6.3.2 Arduino

In order to gauge exactly what it is that students and staff would require from a smart parking system, we will conduct surveys of random people who park on campus. The idea is to discover a general idea of the issues surrounding all parkers, in various parking lots. Therefore, the surveys will be conducted by physically approaching people in parking lots on campus, either as they arrive on campus or are leaving. We aim to have at least 10 survey respondents per parking lot that we survey.

The Arduino development segment of the project will begin with several stages of iterative development. These will contain improvements to the interaction of the pieces of hardware with the controlling board. In addition, there will be significant time placed on getting the detection algorithm right, so that general objects are not confused as cars.

Following this developmental process, at each stage there will be testing and error/bug checking. This will ensure that the system is able to accurately detect a car. The type of tests that we envisage having to conduct are reliability tests, the accurate detection of cars, as well as false-positive tests, whereby a car is incorrectly adjudged to have entered or exited a parking lot. These tests will be conducted with a prototype of the Arduino system at each stage that we will have tests be run.

Once the Arduino is performing as we expect it to, we will begin the integration of the server with the Arduino. This will involve both iterative work on the server side to enable for calls to be made and then from the Arduino, the respective calls need to be made based on whether a car enters or leaves the parking lot. This information will then be relayed onto the app, at whichever iteration it is at that stage.

The last segment will concern the deployment and physical appearance of the system. We will need to source a casing for the Arduino and its components, or design one ourselves, which will allow it to endure any of the potential environmental

6.3.3 Server

This server will be developed as a team, ensuring that it meets the needs as required by both the physical Arduino as well as the app. By doing so, we allow for easy and effective integration from the beginning of development through to the end.

A server will be chosen that is able to handle several requests per second, considering the server needs to update based on the traffic in and out of parking lots. The majority of traffic through the server however will come from the App, where at each refresh or loading of the app, the Server is contacted for its required information.

6.4 Tools required

6.4.1 App

Xamarin is freely available cross platform app development tool. It uses a C#-shared codebase so that developers can write native Android and IOS apps. We chose this platform to develop the Smartphone app because this means we can build apps for both IOS and android users instead of focusing on one group of users. Mark-up languages for interactive prototype design will be used, UCD is similar to agile development in that the design is iterated – testing is done with end users during the development and that's the main difference.

6.4.2 Arduino

The physical Arduino or Arduino clone, such as an Elegoo "Genuino" would be required as the core microcontroller, from which all external components would be connected and controlled. The specific version model will be the UNO R3, due to both its power as well as its portability and reliability. In order for the system to communicate with the central server, a Wi-Fi module is required and so we have elected to go for an ESP-8266 extendable due to its low cost and reliability for the transmission of data on Wi-Fi, especially at far distances. For external components, a Passive Infrared Motion sensor would be required, to detect the presence of an automobile, or similar object, within a certain range. Once the object has been detected, an HC-SR04 Ultrasonic sensor will be used in order to further verify that it is a moving car and to be able to accurately measure its entering and exiting of the parking lot. Additionally, in order to display the parking information to users, we would need an LED board. The information that is transmitted to the server is then also displayed on this board, in its own, easy-to-read form.

6.4.3 Server

We are going to use the Python Django framework to create the server. The Django framework is a popular web framework that makes it easy to build a fully-fledged website or build just the server of the website. The server is built to receive and process the information sent by the Arduino sensor through a wi-fi modem.

7. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

7.1 Ethics clearance

The project involves users throughout the development phase, from surveys all the way to usability tests of the final product. The users will aid in the design of the application.

Before we recruit users, we will require an expedited ethics clearance from The Human Research Ethics Committee. The users would be told before the start of a session the purpose of the information collected and the application.

7.2 Intellectual property

Xamarin SDK is free with the use of Visual Studio and the applications created from Xamarin are not licensed to any company.

Arduino is open-source under GNU Lesser General Public License (LGPL).

7.3 Legal issues

The decision to provide a parking solution such as this on UCT came as a result of the aforementioned. It is therefore important that we have the necessary permissions and clearances from UCT to test our solution on their campuses, as the success of the project is heavily reliant on the testing and implementation of the system specifically on the UCT campuses and parking. Should we fail to attain the clearance, we would be able to replicate the parking scenario of UCT parking at an off-campus location, whereby we would test and further develop the solution. This would then render the project more of a Proof of Concept, which in time, would be able to be introduced into the UCT parking infrastructure.

8. EVALUATION

8.1 App

The most important part of the parking system for most of the users is the parking application, which is the system's user interface. The user interface in product development is the System for end – users and hence the application is more important. The evaluation of the app and its UI and usability are some of the most integral parts of the whole project.

We are going to use User-Centred Design for the design and development of the app, this means that we will run experiments with potential end-users continuously throughout the project – to get insightful feedback to make useful changes iteratively until we have a minimal viable product.

The evaluation of the application is mainly qualitative, we are concerned with how users receive the applications; we can get their opinions from user surveys within the app, or running experiments.

The main purpose of the user tests is to test the usability of the app in terms of relevant and useful features, the UI-interactions and feedback on items or features to remove/add or improve.

The user tests are supposed to inform users on:

- Usefulness of the application currently and in the future
- The features, both tangible and intangible, that work well
- The features that should be improved
- Additional features that should be added
- Does the Application display accurate information.

The tests after the final implementation will evaluate the accuracy and validity of the information displayed on the application.

In addition to user testing, we plan to integrate UCD with the Agile development approach. Agile development encourages test-driven development and thus we also plan to conduct tests including:

- Unit testing
- Accuracy test from server
- And testing for bugs and edge cases (known and unknown)

8.2 Arduino

There will need to be a rigorous testing phase for the physical component, in order to determine whether it will be able to operate fully functionally at all times.

The first set of tests to be conducted will concern the detection algorithm for the incoming and outgoing cars. By further developing the algorithm, we will more accurately be able to test the status of each parking lot. This detection algorithm improvement will also minimise the number of errors experienced by the solution.

The second set of tests will be a network traffic test, whereby we will optimise the format of the data sent through to the server in order to ensure there are no timeouts on the server-side. By doing so we eliminate any errors, whether terminating or not, that may hinder the system's performance.

The last set of tests are more environmental and situational tests where the casing of the system as well as all external components will be exposed to a wide range of elemental forces such as wind and rain. By passing these tests, the final confirmation is given to the solution on it being an easily implementable system with minimal consideration for pre-existing infrastructure. This will again emphasise the portability of the system.

One additional set of evaluation criteria will be to determine the quantitative statistics with regards to the time and frequency of parking on campuses, which could then be displayed to the users in the form of an expected availability in parking lots, based on previous parking behaviour.

9. SUCCESS CRITERION

The main output for this research project should be a fully functioning parking system that can be deployed in one of the most used parking bays on UCT Campus, most likely either P5 or P6 on Upper Campus. The idea is then that this would be extendable to be able to be deployed to any and all parking lots, with minor configurations for each parking lot. The parking system individually includes an Arduino sensor and a mobile application. The Arduino sensor and mobile application should work together fluidly with the help of a server that receives and processes information from the sensor and stores the information to a database. The mobile device retrieves the information from the server and displays it to guide users.

9.1 App

The mobile device needs to be able to synchronise the data from the sensor very quickly and more frequently. The app should meet users' needs, the app should have features that users will use and not have redundant and irrelevant features that will not be used.

The app should be able to inform users whether there is parking in a certain parking bay, it should also be able to recommend a parking bay based on utilisation.

9.2 Arduino

The system should be able to accurately detect the traffic into and out of parking lots with the necessary sensors. The driving habits of students on how they enter and exit the parking lots need to be taken into account as the accuracy of the system as a whole is highly dependent on measuring the traffic of parking lots exactly.

9.3 Server

The server will need to be able to allow for several posts every few seconds, when a car enters or leaves the parking lot. Furthermore, at an even higher rate, the server should allow for several requests, possibly multiple-per-second, from users of the app who would like to determine where there might be parking.

10. PROJECT PLAN

10.1 Risks

The Risk Matrix can be found in Appendix C1, namely, Risks.

10.2 Timeline

A Gantt chart can be found in the Appendix, as C2. This includes all milestones, both individual and group-based.

10.3 Deliverables

Below can be found a list of the deliverables that are required for the project, in chronological order:

23 May	Initial proposal
27 May	Proposal presentations
10 June	Revised proposal
27 June	Paper scaffold
27 June	Survey Results
19 July	Software feasibility demo
19 July	Background/Theory section completed
19 July	Paper prototype
25 July	Interactive prototype
29 July	Final Prototype (MVP)
02 August	Completion of testing and final implementation
16 August	Draft of final paper
26 August	Project paper submission
02 September	Code submission
16 September	Project Demo
23 September	Poster due
30 September	Web page

7 October	Reflection paper

11. WORK ALLOCATION

Joshua will design the Arduino sensor and potentially the LED board for the parking solution.

Thabo will design the mobile application.

The Server will be designed and coded by both Thabo and Joshua, with both of us having key focus of integration into our particular parts of the project.

12. REFERENCES

[1] Redström, J. 2006. Towards user design? On the shift from object to user as the subject of design. *Design Studies*. 27, 2 (2006), 123-139.

[2] Sullivan, L. 1896. The tall office building artistically considered. *Lippincott's monthly magazine*. 403-409.

[3] Khanna, A. and Anand, R. 2016. IoT based smart parking system. *2016 International Conference on Internet of Things and Applications (IOTA)*. (2016), 266-270.

[4] Grazioli, A. et al. 2013. Collaborative Mobile Application and Advanced Services for Smart Parking. *2013 IEEE 14th International Conference on Mobile Data Management*(Milan, 2013), 39-44.

[5] Kopecky, J. and Domingue, J. 2012. ParkJam: crowdsourcing parking availability information with linked data. *The Semantic Web: ESWC 2012 Satellite Events* (Heraklion, 2012), 381-386.

[6] Mahmud, F. and Aris, H. 2015. State of Mobile Crowdsourcing Applications: A Review. *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)* (kuantan, Pahang, 2015), 27-32.

[7] Shoup, D. (2006). Cruising for parking. *Transport Policy*, 13(6), pp.479-486.

[8] Khattak, A. and Polak, J. (1993). Effect of parking information on travelers' knowledge and behavior. *Transportation*, 20(4), pp.373-393.

[9] Choeychuen K. (2013). Automatic parking lot mapping for available parking space detection. *Proceedings of the 5th International Conference on Knowledge and Smart Technology (KST)*. pp. 117–121.

[10] Marquez, M., Lara, R. and Gordillo, R. (2014). A new prototype of smart parking using wireless sensor networks. *2014 IEEE Colombian Conference on Communications and Computing (COLCOM).*

[11] Grodi, R., Rawat, D. and Rios-Gutierrez, F. (2016). Smart parking: Parking occupancy monitoring and visualization system for smart cities. *SoutheastCon 2016*.

APPENDIX

C1. Risk Matrix

Risk	Consequence	Probability	Impact	Mitigation	Monitoring	Management
Coding app takes longer	The deployment of the application would be delayed	Medium	Medium	Following the gantt chart, and keeping track of one another's progress	Break down bigger deadlines into smaller tasks	Break down big tasks into smaller ones and complete them incrementally
Designing Arduino takes longer	There is a delay in the release of the system	Medium	Medium	Begin initial development on time with small increments	Making sure we stick to time and iteratively develop	Start with simple design and build on it iteratively
Scope creep for the app	Delays for all the other steps	Low	High	Add features additional features after the main features	Focus on the features that are most important first	The ranking of requirements will guide us on what to prioritise
Requirements unmet	This will mean that user won't like using the app or sensor	Medium	High	Listening to users needs and keeping the strict deadlines	Feedback from users will guide the requirements effectively	Manage output and deadline expectations with client (supervisor)
Strict estimations	We won't have enough time for a quality software	Medium	Medium	Account for a few hours of unpredictability	Breaking down large tasks into smaller ones for incremental deadlines	Learn from this to design more accurate estimations
One of the members drops out	No integration with sensor and app.	Low	High	Define each members' role or tasks to make them as independent ads possible	Communicate clearly so members know how to manage a missing member's resources	Task distribution, constant communication
Not enough users for testing	Small pool of users means that we won't get an app that address the problems for all	Low	Medium	Design personas and recruit users in advanced. Device affordable incentives	Plan ahead of time with users about their availability for testing	Source more users than needed with the help of the
Users have inaccurate expectations	Delay in project.	Medium	Medium	State clearly to users, what the purpose of the application is	Discuss with users the useful features and novel feature to manage their expectations	Inform users on what the priorities of the application, and why certain features are emphasised over others
Paper prototype is badly received by initial users	Delay in project, spend time creating a better app	Medium	Medium	User surveys before the design process. Having a thorough design process.	Iterative feedback on design, using design principles to guide the design of the paper prototype	Properly understand the feedback, to make the acceptable changes to the interactive prototype
Problems with integration	Delay in final demo.	Medium	Medium	Use reliable technology which is known for its ease in integration.	Ensure development happens iteratively from an integrated example.	Constantly revise our knowledge of both systems so that we both understand for integration.
UCT denies request for testing and deployment on campuses	We will not be able to test the system on campus or deploy it.	Low	High	Apply for permission early on, detailing exactly how it benefits UCT.	In the event that they deny the request, persevere with elevated requests.	In our requests for permission, make it explicit how UCT can benefit now and in the future.
Pre-determined sensors are not sufficient	The initial design and components will have to change.	Low	Medium	Ensure that research done up till now on sensors is sufficient.	At every iteration for the Arduino, ensure the components work	Conduct research on how to manage current sensors as well as exploring others.
The LED display board is stolen	The system would have to exist solely through the app.	Low	Significant	This is primarily a human error, possibly a warning sign included.	Continuously revise the decided location to place the board.	Closely monitor the board once it is deployed.
The Arduino and overall hardware breaks	The entire system would fall apart and be rendered useless.	Low	Catastrophic	Design and all components should be extensively tested under all conditions.	Continuously conduct tests on the hardware that it might be subject to.	Have spare components or Arduino's on hand if there is an issue with one.

C2. Timeline (Gantt Chart)